UTRECHT UNIVERSITY

Department of Information and Computing Science
Department of Physical Geography

Applied Data Science Master Thesis

A Universal Approach: Utilizing Machine Learning to Mimic the Dynamics of Large-Scale, High-Resolution Numerical Models of Geographical Systems

First examiner:
Prof. Dr. Derek Karssenberg

Candidate:
Rick ten Eikelder (BSc.)

Second examiner:
Oriol Pomarol Moya (MSc.)

Cohort:
2022-2023

30^{th} June 2023

Abstract

Complex Earth System processes are commonly studied in the field of geosciences using computationally intensive models. These models often require significant computational resources, resulting in long simulation times. One proposed solution is surrogate modelling, where a part of the complex simulation model is replaced by a Machine Learning (ML) model. In this study, the feasibility of using a transparent Random Forest (RF) model as a universal surrogate for different numerical models of geographical systems is evaluated. The performance of the RF model is assessed in terms of its ability to replicate various pixel-based numerical models, its interpretation capabilities, and its generalisation to unseen data. The accuracy of the results differ significantly between simulation models. The model used in this research is capable of emulating simple simulation models but performs poorly on more intricate models. However, the results demonstrate that the RF model effectively emulates geographical models. With this research a first step is made to construct a general software framework to a universal approach of using ML-models as surrogates models in simulations of geographical systems.

Contents

1. Introduction

In the field of geosciences, computationally intensive models are used extensively to study complex phenomena and simulate various Earth System processes. These models play a crucial role in understanding geological formations, weather patterns, climate dynamics, and other intricate systems [1]–[4]. With the advancement of computational power, geoscientists increasingly rely on numerical calculations to approximate the behaviour of these phenomena. However, it is important to recognise that there are inherent limitations imposed by the available computing resources. The computational time required for these models increases significantly with larger extent, higher resolutions, and increased intricacy of the models. For instance, fully-coupled mid-resolution global climate models, which are used in geoscience, may require weeks to run on supercomputers [5]. This example underscores the substantial computational demands of such models, highlighting the need to consider the available computing resources and the reduction of computation time. For example, in forest fire models the computation time roughly follows $O(n)^3$ where n is the number of cells of one dimension of the grid, assuming a square simulation space [6].

There are many different approaches to reducing computation time without the loss of resolution or scale. One of these approaches is to implement machine learning (ML) models to take over various parts of the simulation model. If the simulation model is coupled with the ML model this construction is called a hybrid model [7], [8]. There are studies where ML is used to downscale spatial and temporal aspects of the models [9]–[12]. Integration of ML models in physics-based models is a growing trend within the field of Earth science [7]–[18].

Over the past few years, the implementation of ML algorithms to support large numerical models has gained significant popularity. This rise in popularity has led to a proposal advocating for the standardised use of ML in climate simulations [19]. However, there are differing views among scientists regarding the integration of ML models to supplement climate models. Some argue that the validity of the model is compromised since the ML model fails to represent the physical processes it predicts [20]. Nevertheless, studies have demonstrated that ML methods can accurately approximate the underlying equations governing the model output data of the simulation model it replaces [21], [22]. It is important to note that the accuracy of ML methods heavily relies on the quality of the data.

To address the challenges associated with computational limitations, it is essential to evaluate the possibilities of using ML to reduce computation time. In this regard, the utilisation of a ML model to function as a surrogate model offers a promising solution. This approach has been performed in several other disciplines with Artificial Neural Networks (ANN) models [23]–[28]. ANN models can accurately represent physics-based patterns, even non-linear patterns. However, the scientific results of these geophysical and climate models are used by politicians

and other policy makers in their decision making process [29]. Therefore, there is a significant need for a transparent decision-making process within the algorithm, allowing for a clear understanding of how predictions are made. The calculations of a NN model quickly become very complex and it is not always clear how the algorithm came to its final result. A Random Forest (RF) model offers more transparency if the structure of the algorithm is kept small. A RF model is considered transparent because it allows for interpretation and understanding of its predictions through the examination of individual decision trees and feature importance.

The primary objective of this research is to evaluate the possibility of utilising a transparent RF model as a universal surrogate for different types of numerical models of geographical systems. An important aspect of this objective is the universality of the approach. This research is a start in the development of a general methodology and software. During this study, the software framework is written as general as possible and allows for any pixel-based simulation to be emulated by the RF model. This objective is divided into the following research questions:

Main question: Can a universal RF model emulate different types of pixel based numerical models of geographical systems?

- How does the RF model perform when it is trained exclusively on the state variable, the variable of interest?
- To what extent is the performance of the RF model dependent on neighbourhood interactions and external input such as precipitation and temperature?
- How effectively does the RF model generalise to unseen data, in terms of interpolation and extrapolation?

To answer these questions, the research starts predicting a single step, applying the RF model to the following simulations: game of life, forest fire model, and snow accumulation model. Subsequently, the aim is to assess the RF model's ability to predict complete simulations using only the state variable from the simulation models. Additionally, the research will explore the incorporation of external input, such as precipitation and temperature, to enhance the performance of the RF model on complete predicted simulations. In order to evaluate the model's generalisability and limitations, the performance on interpolation and extrapolation is analysed. The code can be found on the github repository [30].

2. Methods

This chapter provides a general introduction to surrogate modelling which is followed by the update functions of the simulation models used in this research. For more details on the simulation models see the appendix 6. This chapter continues with details on the RF model and the predictive performance evaluation methods used in this research.The chapter ends with details on the structure and organisation of the data.

2.1 Surrogate modelling

Surrogate modelling provides a valuable framework for emulating simulation models, because of its ability to approximate the behaviour of simulation models while reducing computational demands. By understanding the underlying framework and key concepts, we can better appreciate the applicability and advantages of surrogate modelling in emulating diverse simulation models, such as the snow accumulation simulation and forest fire simulation. The approach to surrogate modelling involves constructing a ML model that aims to emulate the behaviour of the simulation model. By doing so, the surrogate model offers a computationally efficient alternative to running time-consuming forward simulations.

Let us denote the state variables at the current timestep for a specific grid cell as $\mathbf{S}(x, y, t)$, where x and y represent the spatial coordinates and t represents the timestep in the simulation. Let us also denote the external input and parameters at the current timestep for the same grid cell as $\mathbf{I}(x, y, t)$ and $\mathbf{P}(x, y)$, respectively. The update function that calculates the state variables for the next timestep based on the current state variables, inputs and parameters, is described by the following equation:

$$\mathbf{S}(x,y,t+1) = \mathbf{f}(\mathbf{S}(x,y,t), \mathbf{I}(x,y,t), \mathbf{P}(x,y), \mathbf{N}(S(x,y,t))) \ \forall \ x,y,t \ \in \ \mathbf{Q}. \quad (2.1)$$

Where f() represents the update function, $\mathbf{N}(S(x, y, t))$ represents the state variable of the neighbouring cells at timestep t and $\mathbf{Q}$ is the set containing all possible combinations ofx, y and t. This factor is included to account for the spatial dependencies and interactions between adjacent cells. This allows for a more comprehensive and realistic representation of the system dynamics and evolution over time. The surrogate model can only receive the same inputs as the simulation model it emulates. Therefore, the surrogate model can be described by the same structure, where the only difference is that the update function $\mathbf{f}$ is replaced by the ML model $\mathbf{Z}$,

$$\mathbf{S}(x,y,t+1) = \mathbf{Z}(\mathbf{S}(x,y,t), \mathbf{I}(x,y,t), \mathbf{P}(x,y), \mathbf{N}(S(x,y,t))) \ \forall \ x,y,t \ \in \ \mathbf{Q}. \quad (2.2)$$

The ML model predicts the state variable $\mathbf{S}(x, y, t+1)$ one timestep ahead. This prediction, now $\mathbf{S}(x, y, t)$, the neighbourhood information $\mathbf{N}(S(x, y, t))$, the input

I(x,y,t) and parameters **P**(x, y) form the data on which the next prediction is made.

2.2 Simulation models

In this subsection, the simulation models employed in this study are described. The primary focus of this section is to describe the governing equations of the simulations, not on describing the physical understanding of the system represented by these models. The governing equations describe the underlying pattern of the data that the surrogate ML-model is intended to learn. In each subsection the corresponding update function is provided. These update functions are denoted by equations with a similar structure as Eq. 2.1, which serve as a direct representation for the inputs required by the surrogate model to emulate that specific simulation model. More detailed information about the simulation models can be found in Appendix 6.

2.2.1 Cellular automata - (game of life, forest fire)

The game of life and forest fire simulations used in this study are based on cellular automata. Therefore, a short general overview of cellular automata (CA) is described. In a CA model, space and time are discrete and the interactions are purely local. The spatial element in CA is a 2-dimensional grid, where each position in the grid (cell) takes a value describing the state of the system at time t, **S**(x, y, t). During all simulations in this study, the neighbourhood **N**(S(x, y, t)) is defined to be all of the adjacent cells of each cell at (x,y,t) see Figure 2.1 for an illustration. Where the neighbourhood of the target cell (circled) is defined to be all of the cells directly adjacent. The states of all cells are updated by a system that can be represented by the simpler simulation model equation (2.1). In CA models each cell can take one

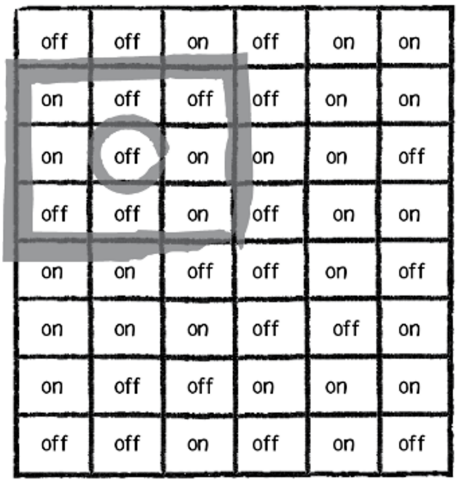


Figure 2.1: Schematic overview of a neighbourhood as defined in this study [31].

of multiple states, in the CA simulations employed in this study each cell can either have the value 0 or 1. For the game of life simulation the states 0 and 1 mean that

the cell is dead or alive, respectively. For the forest fire simulation the states 0 and 1 mean that the cell is 'OFF' or 'ON fire', respectively. The transition from one state to another is determined by a transition rule [32].

2.2.2 Game of Life simulation

In the game of life simulation the state variable $\mathbf{S}(x, y, t)$ can either be 'alive' = 1 or 'dead' = 0. The value for the state variable is determined by the number of neighbours that are 'alive'. See appendix 6 for a more detailed description of the simulation. The update function for the game of life simulation is described by Eq. 2.3,

$$\mathbf{S}_g(x, y, t+1) = \begin{cases} 1, & \text{if } \sum \mathbf{N}(S(x, y, t)) = 2 \text{ or } 3 \\ 0, & \text{Otherwise.} \end{cases} \tag{2.3}$$

Where $\mathbf{S}_g(x, y, t+1)$ represents the new state of the cell at time step t+1, $\sum \mathbf{N}(S(x, y, t))$ counts the alive neighbours. The update function follows the general form of Eq. 2.1 without any input $\mathbf{I}(x, y, t)$ or parameters $\mathbf{P}(x, y)$.

2.2.3 Forest fire simulation

Similarly to the game of life simulation, the state variable $\mathbf{S}(x, y, t)$ is one of two states, 'ON fire' or 'OFF'. In contrast to the game of life simulation, which is fully deterministic, the forest fire model contains a stochastic component. The probability of a cell changing state to 'ON' is calculated in the simulation, see appendix 6 for a more detailed description on how this is calculated. The update function of the forest fire model is described by Eq. 2.4,

$$\mathbf{S}_f(x, y, t+1) = \begin{cases} 0, & \text{if } p_{burn} < u \text{ and } \mathbf{S}_f(x, y, t) = 0 \\ 1, & \text{Otherwise} \end{cases} \text{ for } u \in U. \tag{2.4}$$

Where $\mathbf{S}_f(x, y, t+1)$ represents the state variable, p_{burn} represents the probability of the cell to change state to 'ON fire' and U is the uniform probability distribution on the interval [0-1]. The state variables of the neighbourhood, the elevation and the parameter values for α and $\mathbf{R}$ are explicit requirements for the calculation of p_{burn}.Thus, for emulating the forest fire model,$\mathbf{S}(x, y, t)$, $\mathbf{N}$(S(x, y, t)), elevation as input $\mathbf{I}(x, y, t)_{elevation}$ and the parameters $\mathbf{P}(x, y)$ α and $\mathbf{R}$ are used to train the RF.

2.2.4 Snow accumulation model

In the snow model the state variable $\mathbf{S}(x, y, t)$ represents the amount of accumulated snow on each cell in meter. This is calculated in various steps, more details on this can be found in the Appendix 6. The update function of this simulation model is

described by Eq. 2.5,

$$\mathbf{S}_s(x,y,t+1) = \begin{cases} \mathbf{S}_s(x,y,t) + Snowfall - Snowmelt, & \text{if } Snowmelt < \mathbf{S}_s(x,y,t) \\ Snowfall, & \text{Otherwise} \end{cases} \tag{2.5}$$

Where the conditional statement sets a boundary to prevent the model from generating negative values for snow accumulation. *Snowmelt* and *Snowfall* are calculated based on the precipitation $\mathbf{I}(x,y,t)_{precipitation}$, the temperature $\mathbf{I}(x,y,t)_{temperature}$, the elevation $\mathbf{I}(x,y,t)_{elevation}$ and the temperature lapse rate $\mathbf{P}(x,y)_{\mathbf{TLR}}$. These inputs and parameter are also used to train the ML model.

2.3 Machine learning approach

As mentioned in the introduction, transparency of the model is important when applying a ML model as a surrogate for an important climate model. A RF model is considered transparent because it allows for interpretation and understanding of its predictions through the examination of individual decision trees and feature importance. Therefore, a RF algorithm is chosen which is kept limited in its configuration, to accommodate the transparency. A RF model is transparent because it is build up of a series of binary splits that represents the decision-making process of the model, which can be easily visualised and understood. This interpretability makes it easier to trace and explain the reasoning behind the model's predictions.A RF is an ensemble learning algorithm that combines the predictions of multiple decision trees to make more accurate predictions. See Figure 2.2 for an example of part of a decision tree used to model the game of life simulation. The RF model operates by constructing multiple decision trees during the training phase and averaging their outputs to obtain the final prediction. At each decision the gini index is optimised, the algorithm optimises for the highest level of homogeneity in the data after the split. The samples used to make each decision are sampled from the data set available at the first node. In this study bootstrapping is used, this entails that at the first node of a tree a random subset of the total available data is chosen [33]. Each decision tree is then trained on one of these subsets using a random selection of features. By introducing randomness in both the data and feature selection, the RF model reduces the risk of overfitting and improves generalisation.

2.3.1 Random forest configuration

In designing the RF model, the trade-off between accuracy and transparency is taken into account. With the goal in to make a first step towards a universal surrogate ML model for geographical systems, the RF configuration is unchanged throughout the study. The RF configuration is limited to contain a maximum of 10 trees, each with a maximum depth of 10 decision layers. This limitation of the number of trees and the depth helps prevent excessive complexity and potential

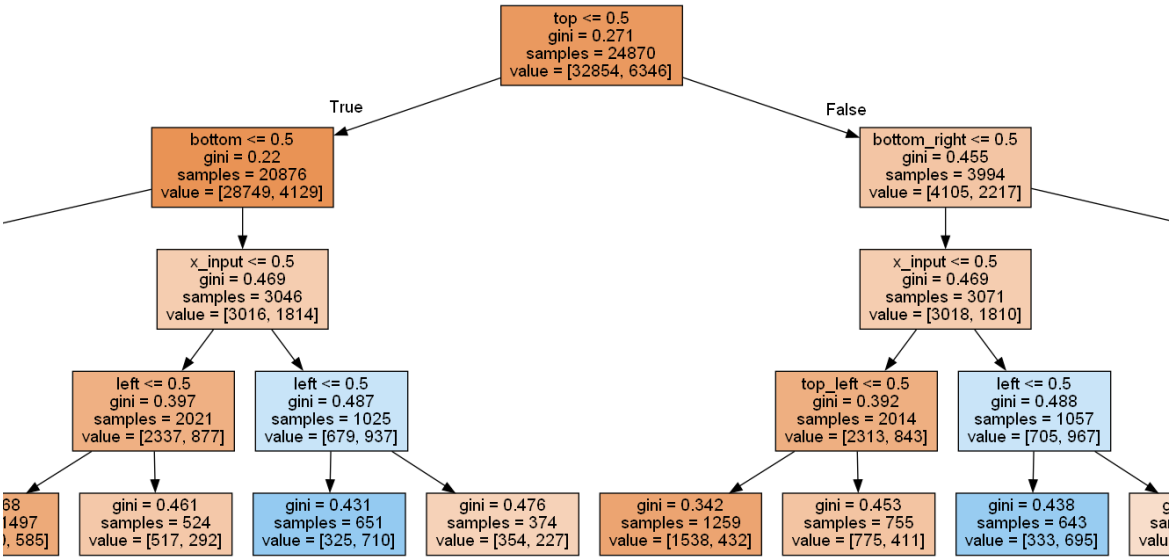


Figure 2.2: Part of a possible decision tree for emulating the game of life simulation. Each rectangle represents a decision. The arguments in each rectangle are from top to bottom: decision criteria, impurity of data index, number of samples used to make this decision, number of samples. The terms in the image such as 'bottom', 'left' refer to the neighbours of the cell.

overfitting, making the model more transparent and easier to interpret. For the game of life and forest fire simulations a RF classifier model is used. For the snow accumulation model a RF regressor model is used. To summarize, the configuration of the model is kept the same with the following arguments: Maximum trees $(n_{estimators} = 10)$, maximum tree depth $(max_{depth} = 10)$ and include bootstrapping as discussed in the theory section.

2.3.2 Performance evaluation methods

The performance of the RF method is evaluated in different ways for different models. The game of life and forest fire simulations assign a Boolean value to each pixel and can therefore be evaluated by constructing a confusion matrix. From the results of this confusion matrix the corresponding accuracy can be calculated. In the next stages of the research full simulations are predicted. In evaluating the forest fire model, a timeseries of the area of the fire is generated for comparison with simulations. Unlike the snow model evaluation, where tracking the amount of snow provides a clear timeseries, the forest fire model considers cells as 'OFF' until they change state to 'ON'. This information only reveals the timestep when a single cell changes state. However, the forest fire has a stochastic component, and the ratio between α and $\mathbf{R}$ affects the fire's direction.

Predictions on the snow accumulation model are evaluated on the mean squared error (MSE) between the predicted amount of snow and simulated amount of snow on each pixel over the complete time of the simulation. The MSE is a commonly used and easy to interpret metric. This generates a map with the MSE of each pixel.

The snow accumulation model is also evaluated on the timeseries of the amount of snow on 4 points in the simulation, see Fig 2.3.

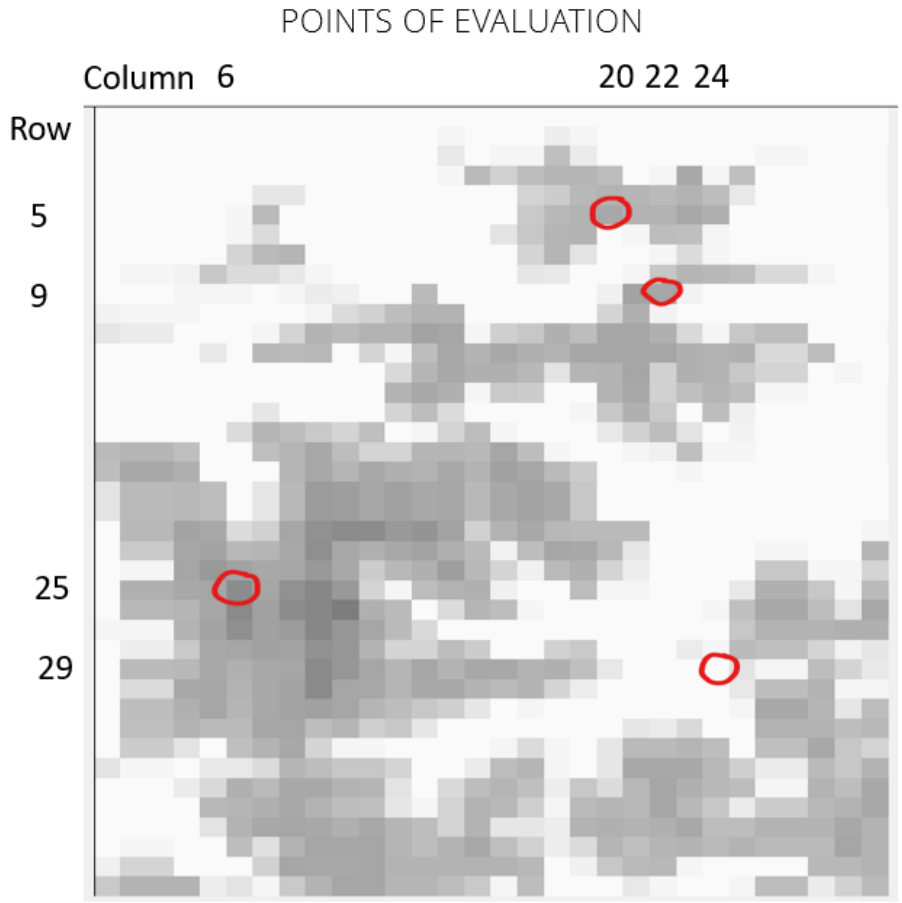


Figure 2.3: Evaluation points for the snow model.

The points are chosen based on the pattern of the amount of snow on these points during the simulation. Two of these points showed an increasing pattern in the amount of snow and the other two showed an oscillating pattern of accumulation and complete melting of the snow.

2.4 Data

All data used in this research is generated by simulations. The forest fire simulation, snow accumulation simulations and the data regarding the elevation, temperature and precipitation are provided by the PCRaster tutorial [34].

2.4.1 Training and test data

The target feature is for every simulation at all stages throughout the studies the state variable of the next timestep $\mathbf{S}(x, y, \text{t+1})$. The training data is generated by running the simulation model with different values for the parameter $\mathbf{P}(x, y)$. For the

interpolation and extrapolation studies an extra simulation is run with the parameter that is to be emulated by the RF. This last complete simulation is not contained in the training data and kept separate to test the model predictions. The RF emulates a full simulation step wise, it predicts the state variable for t+1. At each prediction step the model receives the previous prediction $\mathbf{S}(x, y, t)$, possible external input $\mathbf{I}(x, y, t)$, neighbourhood $\mathbf{N}(S(x, y, t))$ and parameters $\mathbf{P}(x, y)$ for the next prediction. This way, at each time step the surrogate model receives the same information that would otherwise be given as input to the simulation model at this timestep. This process is schematically visualised in Figure 2.4. In the proof of concept predictions

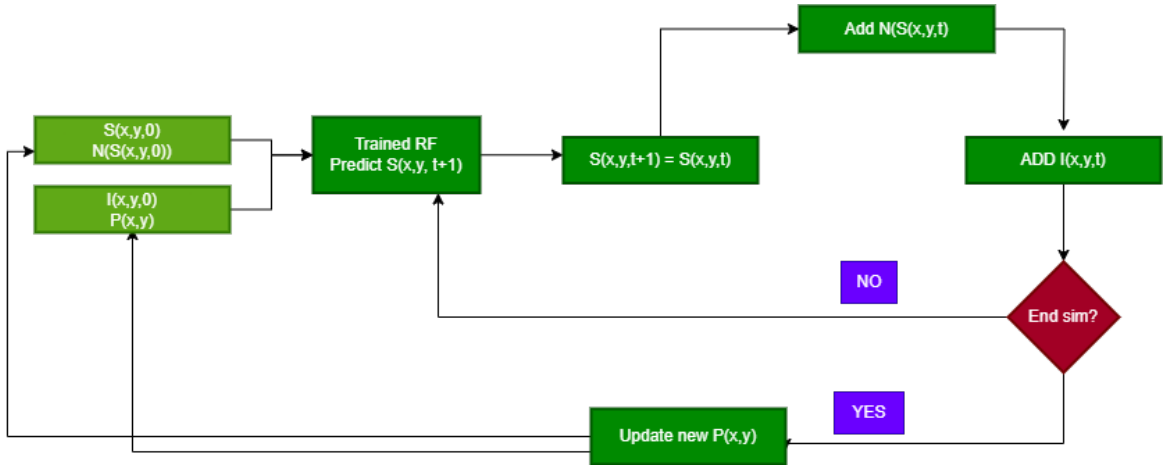


Figure 2.4: A schematic representation of the prediction cycle.

only the state variables $S(x, y, t)$ and neighbourhoods $\mathbf{N}(S(x, y, t))$ are taken into account for training and testing. For the rest of the research the training data is represented in Table 2.1. The testing data always contained one full simulation that was executed with the testing parameter $\mathbf{P}(x, y)$.

Table 2.1: This table shows an overview of which features are taken into account for emulating which model

Models	$\mathbf{S}(x, y, t)$	$\mathbf{N}(S(x, y, t))$	$\mathbf{I}(x,y,t)$	$\mathbf{P}(x, y)$
Game of life	Cell dead = 0 or alive = 1	$\mathbf{S}(x, y, t)$ for each neighbour	-	-
Forest fire	Cell 'ON fire' = 1 or 'OFF'= 0	1 IF any of the neighbours = 1	Digital Elevation Map	**R** α
Snow	Amount of snow in meter	-	Precipitation, Temperature Digital Elevation Map	**TLR**

In the part of the research where the influence of the input $\mathbf{I}(x, y, t)$ is evaluated, the models are first trained on only the state variables $\mathbf{S}(x, y, t)$ so that the prediction is solely based on the previous prediction. These results are then compared to a RF model that is trained on features as described in Table 2.1.

3. Results

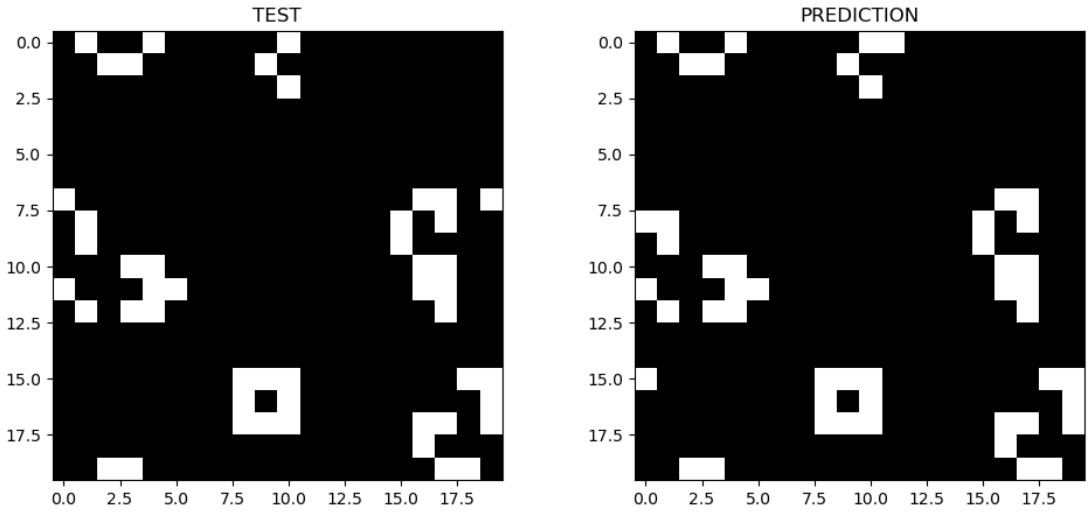
In this chapter the results of the study are presented. Firstly, the universal RF model is tested for one-step prediction across three distinct simulation models: the game of life simulation, a forest fire model, and a snow accumulation model. Subsequently the influence of the external input $\mathbf{I}(x,y,t)$is evaluated. This chapter concludes with the results of the interpolation and extrapolation study.

3.1 Proof of concept: one-step prediction

The results of the one step prediction on the game of life, forest fire model and snow model are described in this chapter.

3.1.1 Game of Life

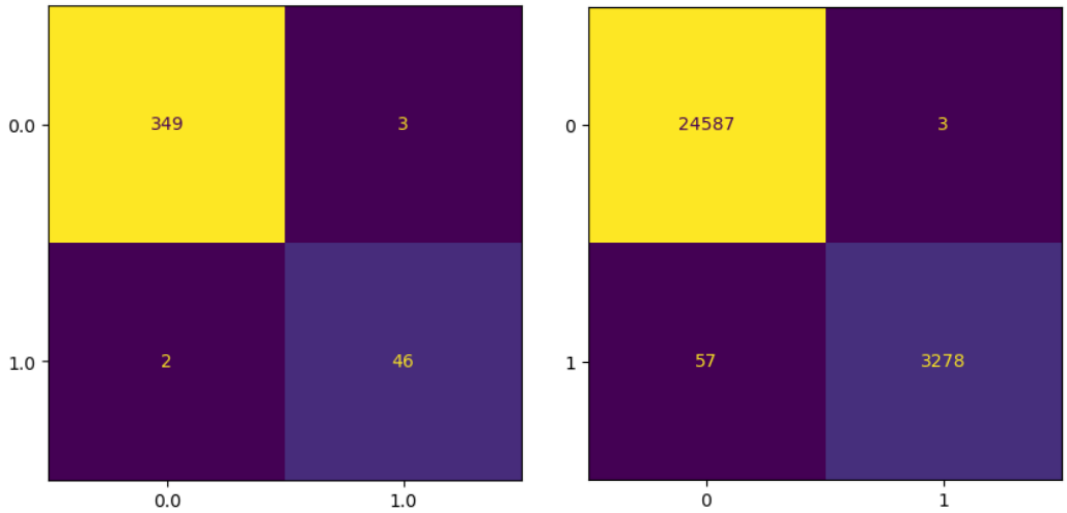
The RF model classifies each cell as either 'alive' or 'dead'. Dead pixels are black and alive pixels are white. In Figure 3.1 the target state is shown on the left and the prediction is shown on the right. The prediction performs well except on the edge of the simulation. In emulating the game of life simulation, the RF is supposed to pick up on the algorithmic relation between a state and it's neighbours as explained in Appendix 6.



(a) Target for the prediction. **(b)** Prediction

Figure 3.1: Comparing the target state and the predicted state for a one-step prediction of a game of life simulation. White squares are 'alive' and black squares are 'dead' in the simulation. The left figure shows the target and the right figure shows the prediction.

The wrong classifications at the edge are an artefact of a differently defined neighbour in the simulation then in the data preparation for the ML model. This method for determining neighbours is only used at the game of life simulation. However, this problem is reduced as the simulation space increases. The ratio of the edge to the total amount of pixels is $\frac{4x}{x^2}$, where x is the size of one edge of the simulation space in a square simulation. Overall the RF model performs well as can be seen in the confusion matrix in Figure 3.2a. The accuracy of the prediction is 0.987.



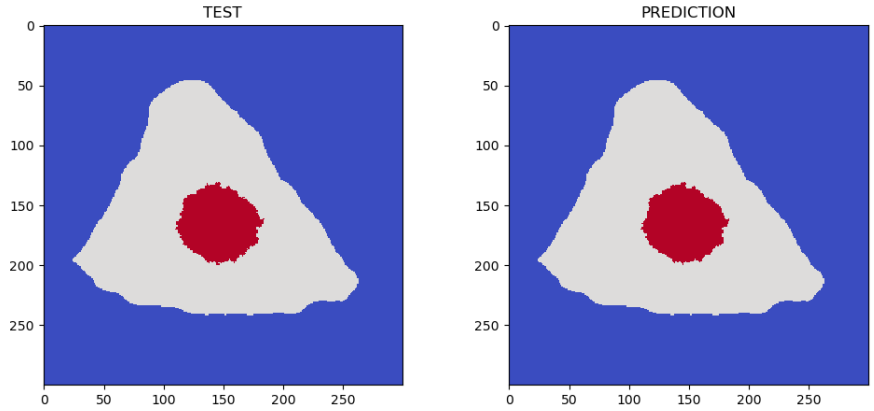
(a) Confusion matrix of the prediction of the game of life simulation. Where 0.0 represents a pixel that is 'dead' and 1.0 represents a pixel that is 'alive', with on the vertical axis the target values, on the horizontal axis the simulated values.

(b) Confusion matrix of the prediction of the forest fire simulation. Where 0 represents a pixel that is 'OFF' and 1 represents a pixel that is 'ON', with on the vertical axis the target values and on the horizontal axis the simulated values.

Figure 3.2: Confusion matrices for the forest fire simulation [right] and game of life simulation [left] one step predictions.

3.1.2 Forest Fire

For the emulation of the forest fire simulation, the RF model classifies any pixel on the map to be either 'OFF' or 'ON FIRE'. The result of the one step prediction is shown in Figure 3.3, with on the left the target state and on the right the predicted state. The overall shape of the fire is the same in the prediction and the target. Around the edges of the fire there are some small deviations. The overall performance is shown in the confusion matrix in Figure 3.2b. The accuracy of the prediction is 0.998. This is a good indication that the RF can accurately predict one step of the forest fire simulation.



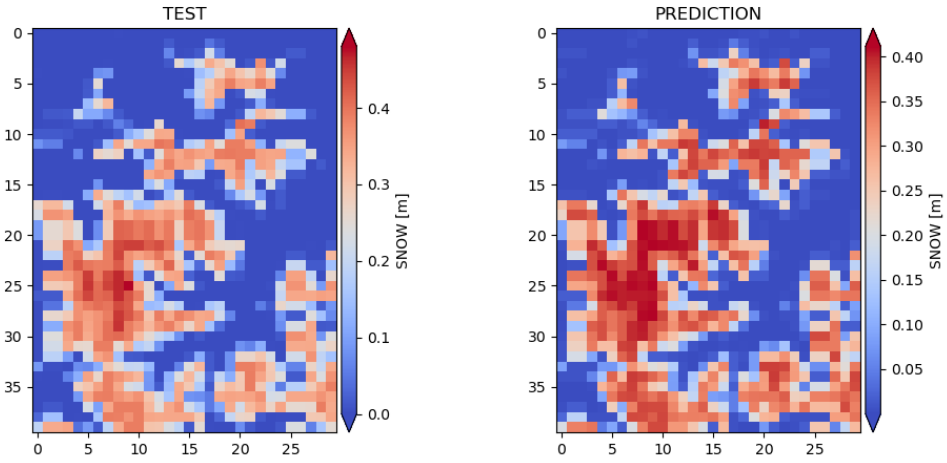
(a) Target for the prediction. **(b)** Prediction

Figure 3.3: Comparing the target state and the predicted state for a one-step prediction of a forest fire simulation. Grey squares are 'OFF', the red squares are 'ON' fire. Blue squares are not on the map.

The values outside of the map (grey), but in the simulated space (blue) are not taken into account in the confusion matrix and accuracy calculation because these would disrupt the accuracy measure. The accuracy is determined only by the pixels on the map.

3.1.3 Snow accumulation model

In contrast to the previous two classification models, a RF regressor model is now constructed to predict the amount of snow. The one step prediction is depicted in Figure 3.4, on the left is the target state and on the right is the predicted state.



(a) Target for the prediction. **(b)** Prediction

Figure 3.4: Comparing the target state [left] and the predicted state[right] of a snow accumulation simulation. The colors on the map illustrate the snow depth in meters for each pixel.

The predicted amount of snow is too high, but the spatial pattern of the prediction approximate the target simulated state well.

3.2 Simulation predictions trained on state variable

The data used to train the RF-model in this section contains only the state variables of the previous step $\mathbf{S}(x, y, t)$.

3.2.1 Forest Fire

To evaluate the evolution of the forest fire the simulated and predicted area of the fire is compared at each timestep, see Figure 3.5. The predicted fire does not increase in size.

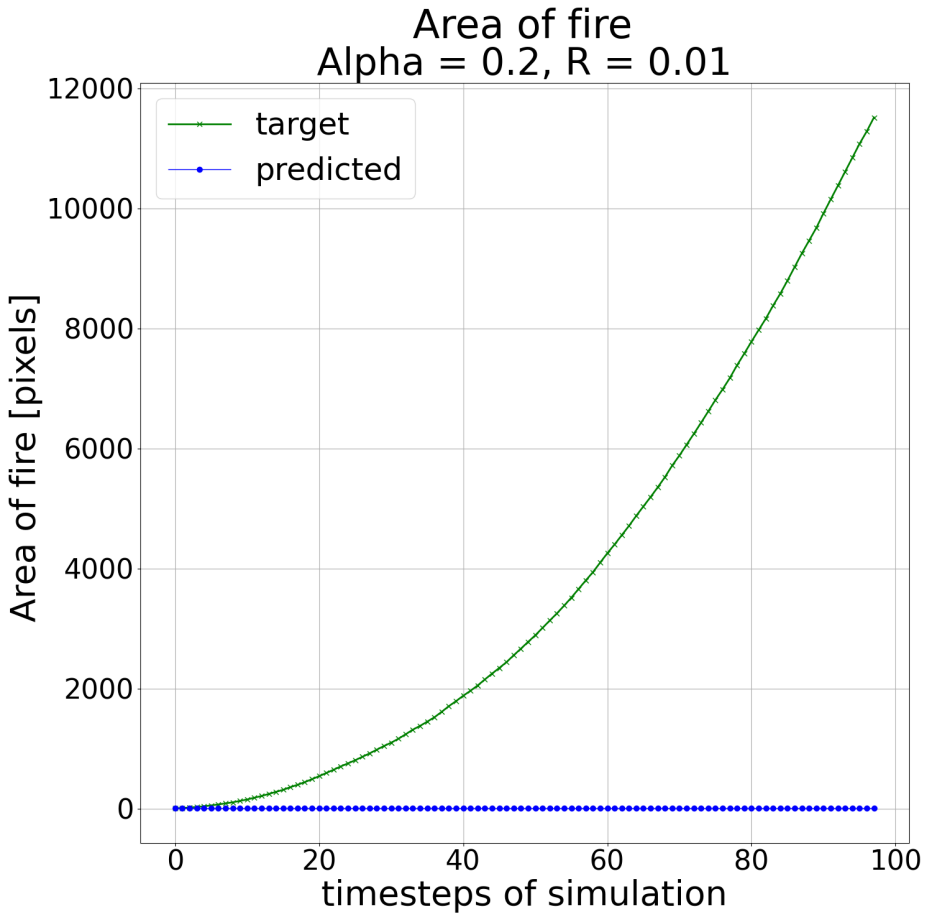


Figure 3.5: The area of the fire per timestep. The green line represents the area of the simulated fire. The blue line represents the area of the predicted fire. In these predictions only the previous state is used as input for the next state.

The ML-model was unable to learn the underlying pattern governing the fire spread in the simulation. This was to be expected, there is no causal link between not being on fire and catching on fire. The simulated fire only spreads by proximity, since burning projectiles are not included in the simulation model.

3.2.2 Snow accumulation model

In this section, the predicted forest fire and snow model are evaluated. The predictions of the model based only on the state variable for the snow accumulation model are shown in Figure 3.6. The amount of snow is evaluated on the 4 points described in the methods section, see Fig 2.3 in the simulated space.

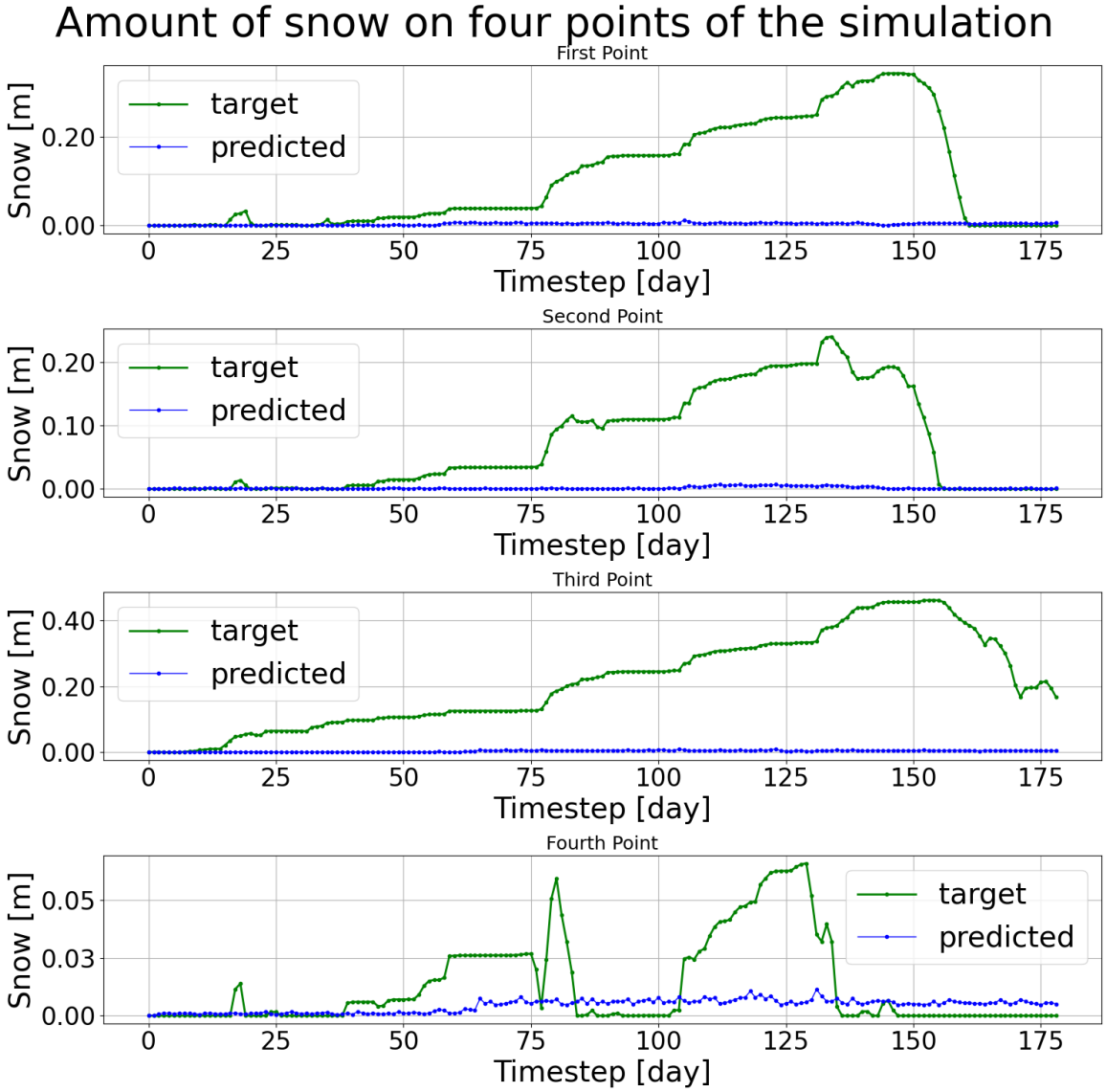
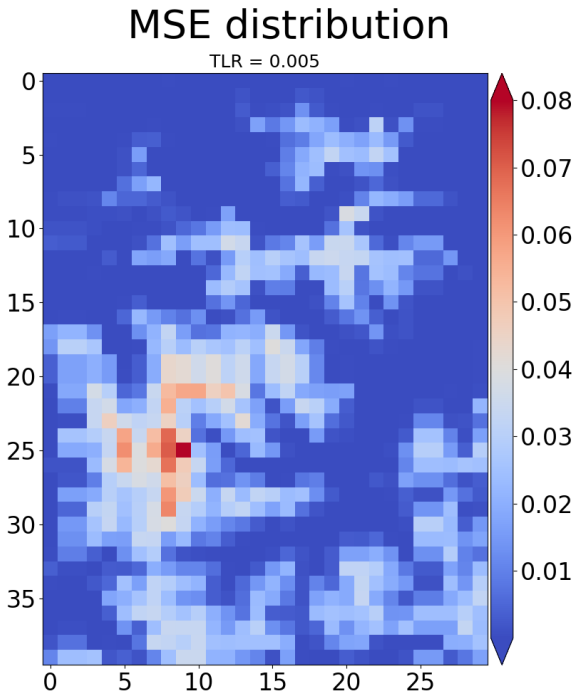


Figure 3.6: The amount of snow calculated on four points. RF model exclusively trained on $\mathbf{S}(x, y, t)$. The green line represents the amount of snow at each timestep of the simulation model, the blue line represents the amount of snow at each timestep of the predictions of the ML-model.

The ML-model did not learn the underlying pattern of the snow accumulation based on the amount of snow in the previous step only. This makes sense since the future amount of snow is not only dependent on the current amount of snow. The predictions of the ML-model are fluctuating slightly, in the timeseries of point

four is visible that there is a relatively small, random deviation from the starting point in the predictions. The predictions do not follow the trend of the simulated timeseries on these points. The error on each pixel, averaged over the timeseries, is shown in Figure 3.7.

Figure 3.7: The mean squared error (MSE) between the predicted and simulated values. The MSE is calculated per pixel averaged over the timeseries.



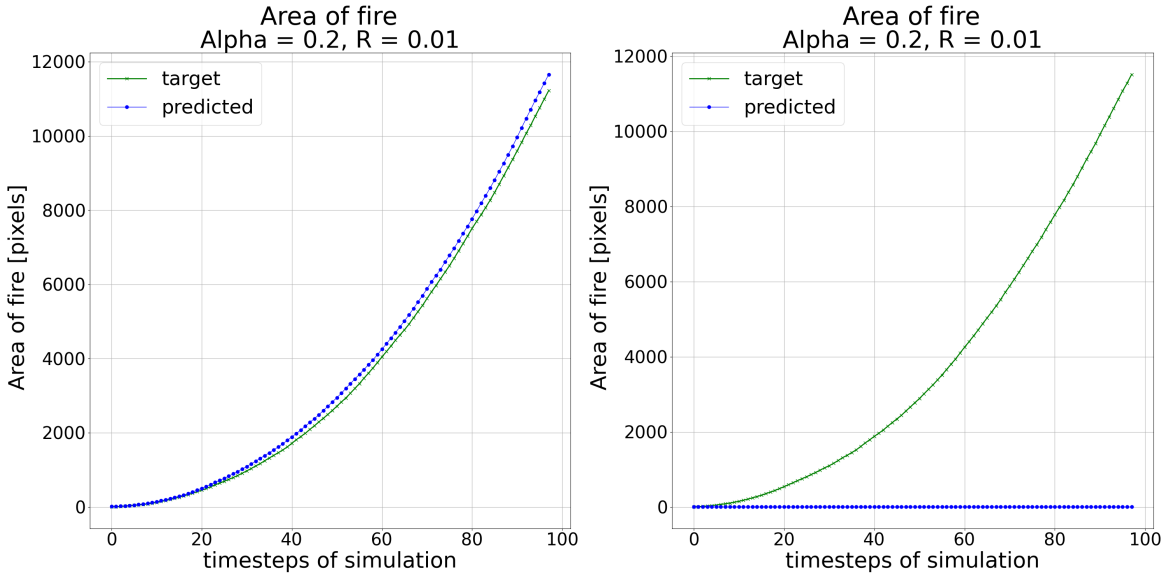
Since the predicted values are practically unchanged over the timeseries, this MSE map can be used as a reference for maximum error for this simulation trained on a **TLR** of 0.005.

3.3 Simulation predictions trained on external input and neighbourhood interactions

In this chapter the results are presented on full simulations including external input, neighbourhood interactions and parameters. The RF model in this section has been trained on the features as described in Table 2.1.

3.3.1 Forest Fire

The results in this subsection are based on the RF model trained on $\mathbf{S}(x, y, \mathrm{t})$, $\mathbf{N}(\mathbf{S}(x, y, \mathrm{t}))$, $\mathbf{I}(x, y, t)_{elevation}$and parameters $\mathbf{P}(x, y)$, α and $\mathbf{R}$. This result is compared to the previous predicted fire spread in Figure 3.8.

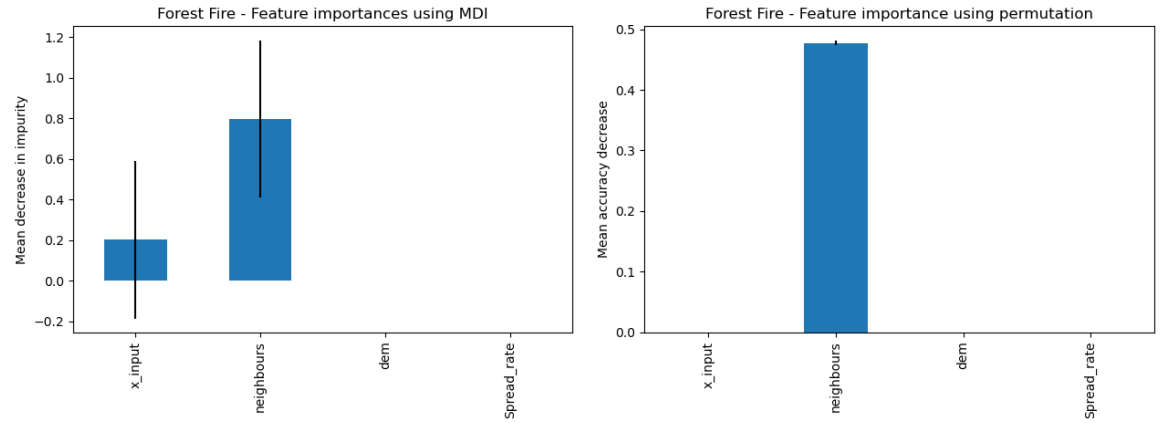


(a) Including neighbouring pixels.

(b) No neighbouring pixels.

Figure 3.8: Compare performance of RF model with and without neighbourhood, parameters and external input. The green line represents the area of the simulated fire. The blue line represents the area of the predicted fire.

The RF model is able to predict the increase in the area of the fire. The area of the predicted fire increases faster then that of the simulated fire. The difference in performance is evident and in line with expectation. The importance of each feature is evaluated to demonstrate the importance of the neighbouring pixels, see Figure 3.9.



(a) Feature importance is computed as the mean and standard deviation of accumulation of the impurity decrease within each tree [35].

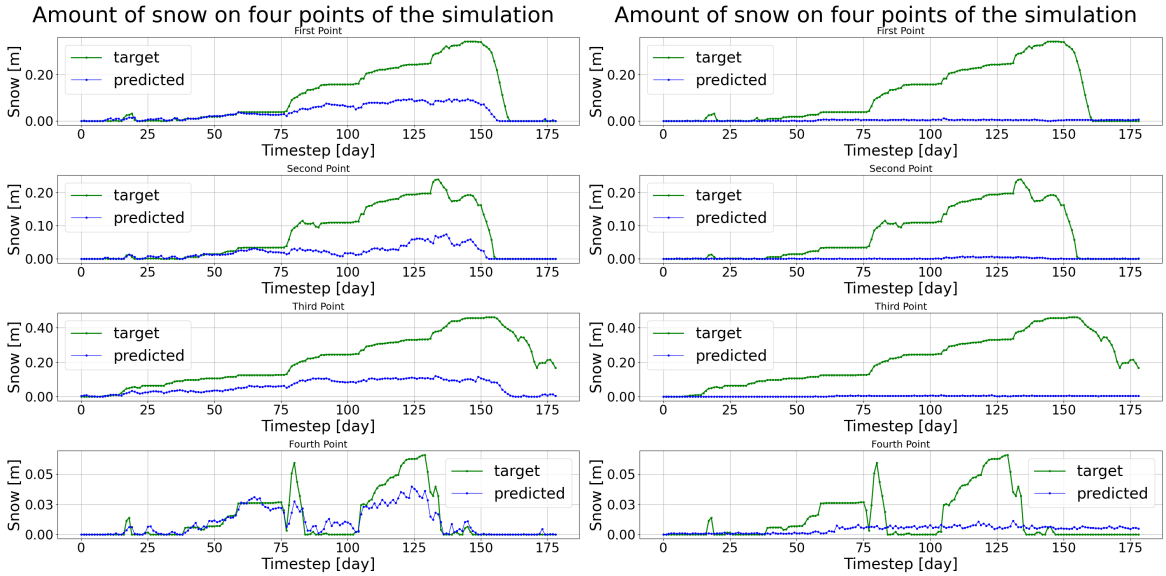
(b) The feature importance is determined based on the model performance by randomly shuffling one feature [36].

Figure 3.9: Feature importance for the RF trained on the forest fire simulation.

The evaluation of feature importance indicates that the contribution of neighbouring pixels to the prediction is substantial. This is in line with expectation since the P_{burn} is explicitly dependent on $\mathbf{N}(\mathbf{S}(x, y, t))$.

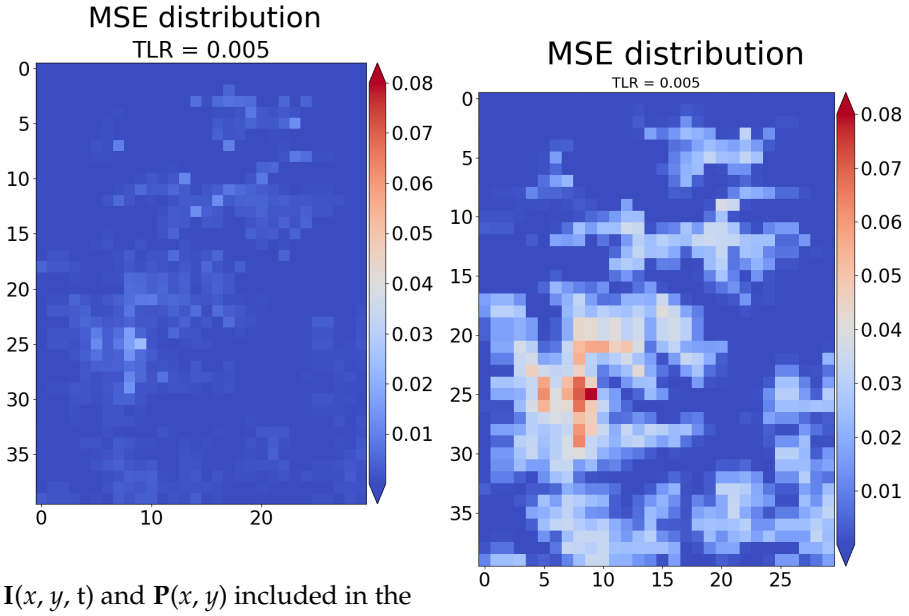
3.3.2 Snow accumulation model

The temperature lapse rate (**TLR**) for the predicted simulation is 5e-3 [°C km^{-1}]. The following results are generated by the RF trained on the parameter **TLR**. The predictions are compared to the predictions of the previous section in Figure 3.10.



(a) Including **I**(x, y, t) and **P**(x, y) in the model. **(b)** Solely trained on **S**(x, y, t).

Figure 3.10: Compare performance of RF model with and without **I**(x, y, t) and **P**(x, y). The amount of snow in meter calculated on four points. The green line represents the target amount of snow, the blue line represents the predicted amount of snow.



(a) $\mathbf{I}(x, y, t)$ and $\mathbf{P}(x, y)$ included in the model.

(b) Model solely includes on $\mathbf{S}(x, y, t)$

Figure 3.11: The MSE of the predictions solely based on $\mathbf{S}(x, y, t)$ [right] is compared to the predictions based on $\mathbf{S}(x, y, t)$, $\mathbf{I}(x, y, t)$ and $\mathbf{P}(x, y)$ [left]. The MSE is calculated per pixel averaged over the timeseries.

The abrupt transitions are accurately represented in the predictions that include $\mathbf{I}(x, y, t)$ and $\mathbf{P}(x, y)$, as seen in Figure 3.10a. There is a significant discrepancy between the predicted amounts of snow and the target. The error of these predictions is compared to the baseline error in Figure 3.11. The map with the MSE of the predictions including the external input and parameters, is shown in Figure 3.11a. The error is lower over the entire map for the model that included $\mathbf{I}(x, y, t)$ and $\mathbf{P}(x, y)$. Notably, the highest error is lower, approximately 0.035 compared to 0.08. The spatial pattern is similar in both predictions.

3.4 Interpolation and extrapolation analysis

This section demonstrates the performance of the RF model in predicting simulations with interpolated/extrapolated parameters $\mathbf{P}(x, y)$. The forest fire emulations are tested on α and $\mathbf{R}$. The snow accumulation model is tested on $\mathbf{TLR}$.

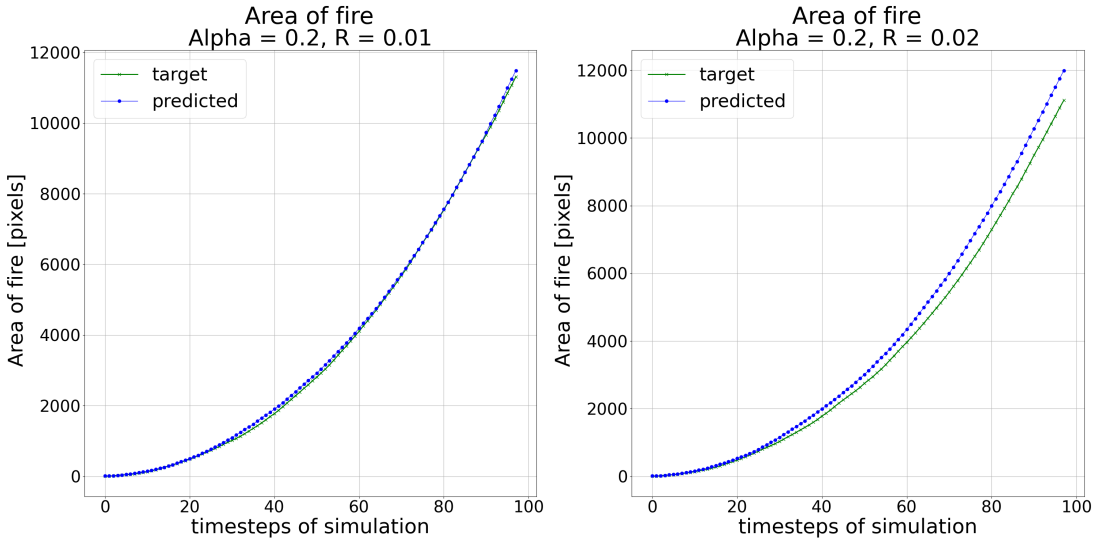
3.4.1 Interpolation

The predictions in this subsection emulate a simulation of which the $\mathbf{P}(x, y)$ lies within the range of values for this parameter used in the training data.

Forest Fire

The forest fire simulation is tested on interpolation for both α and $\mathbf{R}$. The results for α and $\mathbf{R}$ can be seen in Figure 3.13 and Figure 3.12, respectively. The evaluation

of the performance on interpolated values of **R** involves maintaining a constant α during both the training and testing phases, here α = 0.05. The outcomes are presented in Figure 3.12. This simulation is divided into two segments, with Figure 3.12a illustrating the prediction of the simulation with **R** = 0.2, while Figure 3.12b showcases the prediction of the simulation with and **R** = 0.3.

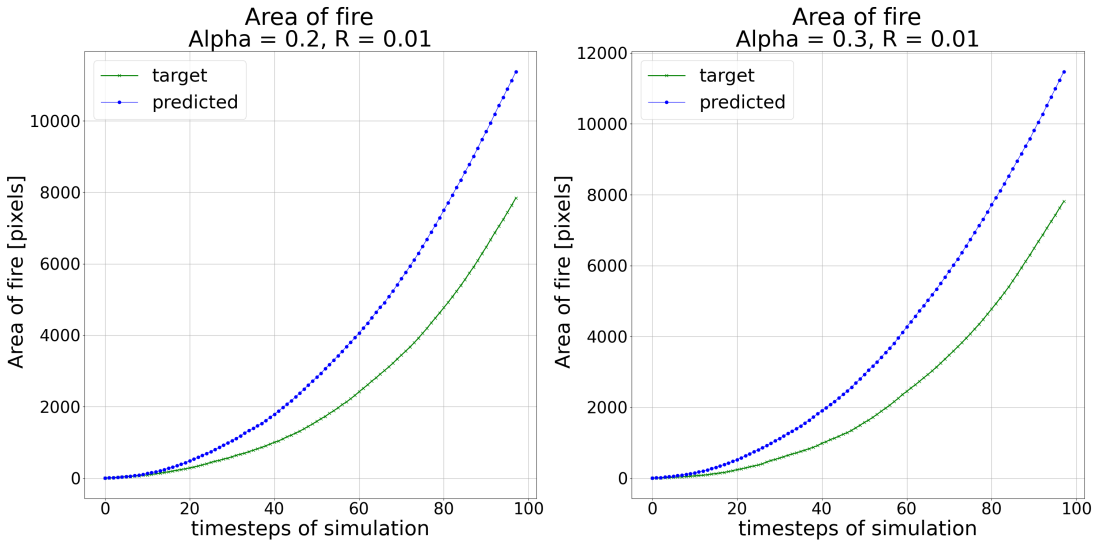


(a) Train: **R** = [0.005, 0.015, 0.02, 0.025, 0.003, 0.035]
Test: **R** = [0.01]

(b) Train: **R** = [0.005, 0.01, 0.015, 0.025, 0.003, 0.035]
Test: **R** = [0.02]

Figure 3.12: Constant α = 0.05. Interpolation of R. The green line represents the area of the simulated fire. The blue line represents the ML predictions.

For both values of **R**, the RF model successfully predicts the curve for the first 30 predictions, with the prediction for **R** = 0.01 being significantly more accurate then the previous result shown in Figure 3.8a. However, around the 30th predictions, the model starts to overestimate the rate of the fire spread for **R** = 0.02. The evaluation of the performance on interpolated values of α involves maintaining a constant **R** during both the training and testing phases, here **R** = 0.01. The outcomes are presented in Figure 3.13. This simulation is divided into two segments, with Figure 3.13a illustrating the prediction of the simulation with α = 0.2, while Figure 3.13b showcases the prediction of the simulation with α = 0.3. The RF model appears to be less effective in interpolating predictions of α. For both values of α the model overestimates the growth rate of the fire. The underperformance of the model on interpolation of α becomes clear when comparing Figure 3.13a with Figure 3.12a. Both final simulations are performed with the values α = 0.2 **R** = 0.01. The training data for different values of α does not represent target simulation accurately.



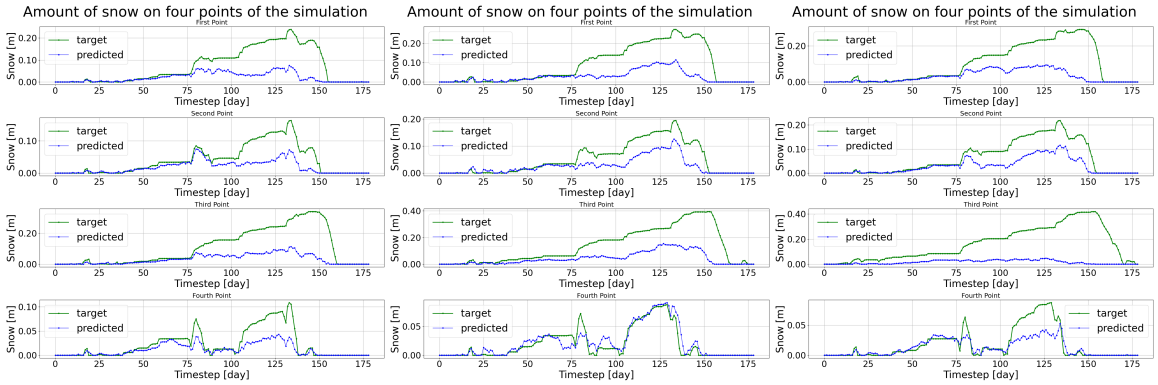
(a) Train: α= [0.05, 0.1, 0.15, 0.25, 0.3, 0.35]
Test: α = [0.2]

(b) Train: α = [0.05, 0.1, 0.15, 0.2, 0.25, 0.35]
Test: α = [0.3]

Figure 3.13: Constant **R** = 0.01, interpolation of α. The green line represents the area of the simulated fire. The blue line represents the ML predictions.

Snow accumulation model

In general it is clear that the predictions follow the pattern of the simulated snow accumulation, as seen in Figure 3.14. The patterns between the prediction and target of the individual points fits better compared to Fig **??**.



(a) Train: **TLR** = [0.001, 0.003, 0.004, 0.005]
Test: **TLR** = [0.002]

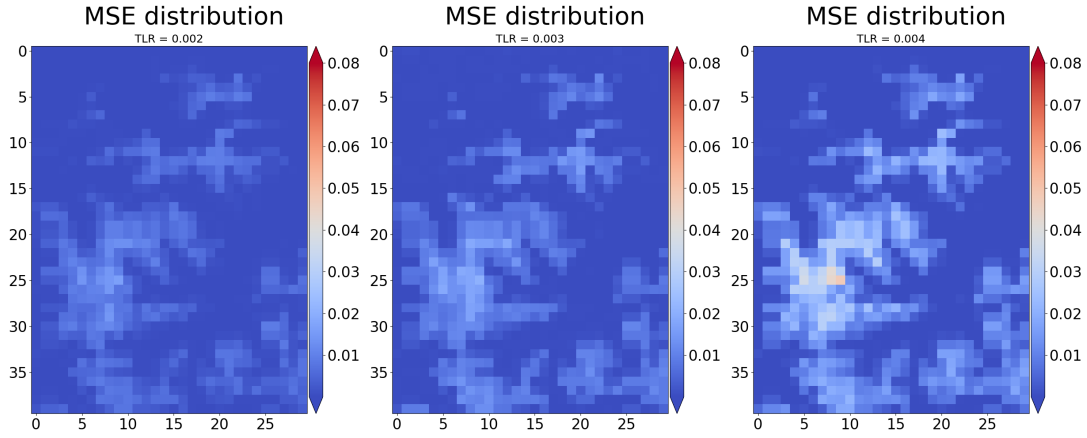
(b) Train: **TLR** = [0.001, 0.002, 0.004, 0.005]
Test: **TLR** = [0.003]

(c) Train: **TLR** = [0.001, 0.002, 0.003, 0.005]
Test: **TLR** = [0.004]

Figure 3.14: The amount of snow in meter calculated on four points. The green line represents the amount of snow at each timestep of the simulation model, the blue line represents the amount of snow at each timestep of the predictions of the ML-model.

The MSE map of these predictions as can be seen in Figure 3.18. The MSE of all interpolated emulations are below 0.006 max. This is not lower then the previous

best MSE of 0.04 at the predictions of Figure 3.11a. In the simulations with a lower **LTR** the target is lower in general.



(a) Interpolated on **TLR** of 0.002.

(b) Interpolated on **TLR** of 0.003.

(c) Interpolated on **TLR** of 0.004.

Figure 3.15: MSE of predictions on interpolated **TLR**. The MSE between the predicted and simulated values. The MSE is calculated per pixel averaged over the timeseries.

3.4.2 Extrapolation

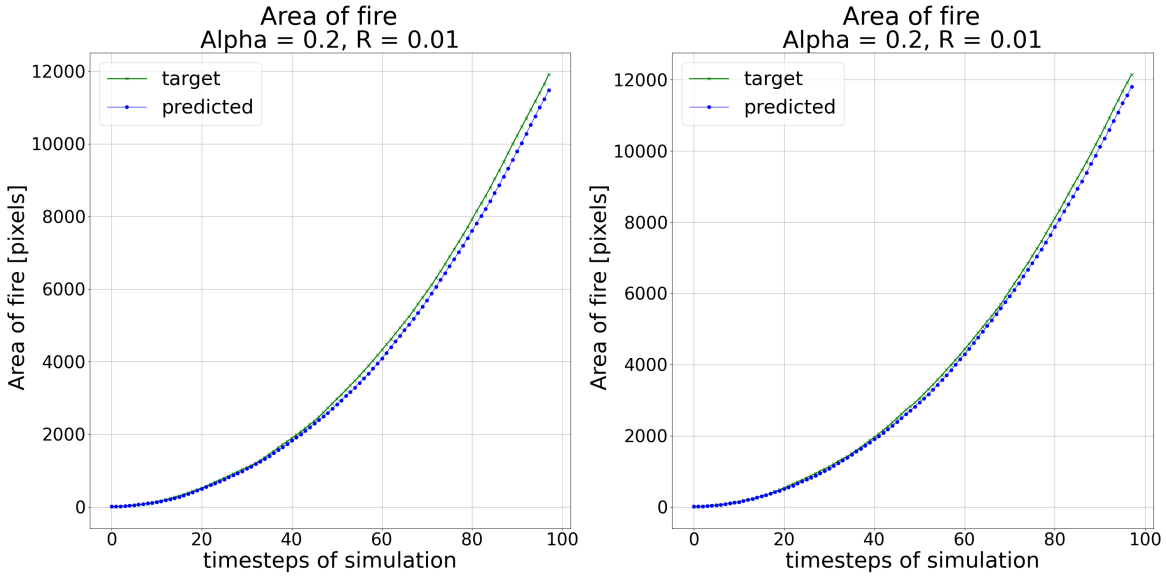
The predictions in this subsection emulate a simulation of which the $\mathbf{P}(x, y)$ lies outside the range of values for this parameter used in the training data.

Forest Fire

The RF emulating the forest fire simulation is tested on extrapolated values of α in Figure 3.16a and on extrapolated values of **R** in Figure 3.16b. The evaluation of the performance on interpolated values of **R** involves maintaining a constant α during both the training and testing phases, here α = 0.05. The outcomes are presented in Figure 3.12. This simulation is divided into two segments, with Figure 3.12a illustrating the prediction of the simulation with **R** = 0.2, while Figure 3.12b showcases the prediction of the simulation with and **R** = 0.01.

Snow accumulation model

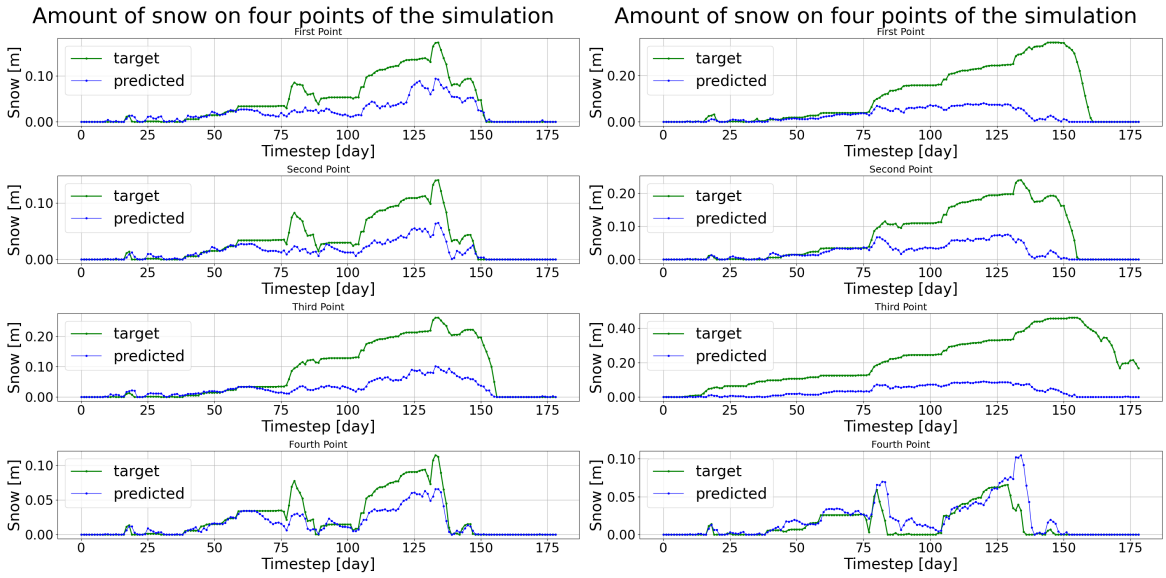
Extrapolation of **TLR**. It is visible in Figure 3.17 that the extrapolation for low rates of **TLR** are slightly better then the extrapolation for higher rates. However, in both cases the emulation unperformed.



(a) Constant **R** = 0.01.
Train: α = [0.25, 0.3, 0.35, 0.4, 0.45, 0.5]
Test: α = [0.2]

(b) Constant α = 0.2.
Train: **R** = [0.02, 0.03, 0.04, 0.05, 0.06, 0.07]
Test: **R** = [0.01]

Figure 3.16: Extrapolation of α (left) and **R** (right). The green line represents the area of the simulated fire. The blue line represents the ML predictions.



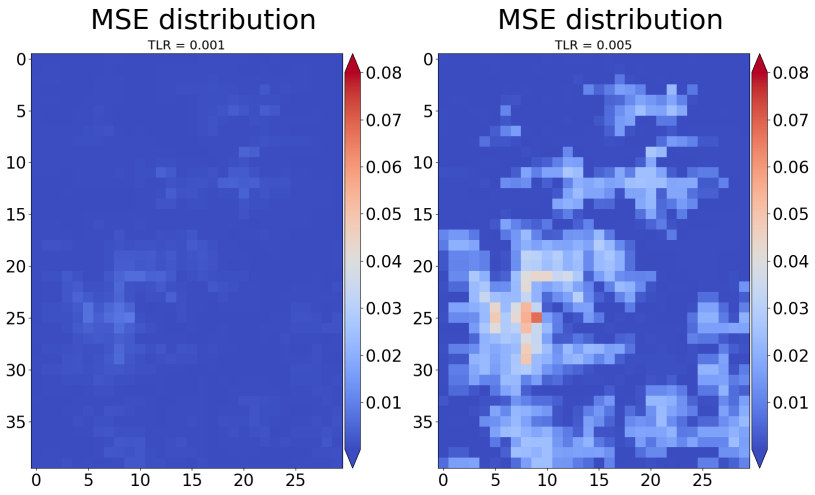
(a) Train: **TLR** = [0.002, 0.003, 0.004, 0.005] Test: **TLR** = [0.001]

(b) Train: **TLR** = [0.001, 0.002, 0.003, 0.004] Test: **TLR** = [0.005]

Figure 3.17: Extrapolated results of the **TLR** of the snow accumulation model. The green line represents the amount of snow at each timestep of the simulation model, the blue line represents the amount of snow at each timestep of the predictions of the ML-model

The MSE map of the lower **LTR** is much lower then the MSE of the higher rate. However, this is not necessarily due to a better prediction. The target values of the lower rate are much lower since the temperature drops less on higher altitudes,

there is less snow on each pixel.



(a) Extrapolated on **TLR** of 0.001. **(b)** Extrapolated on **TLR** of 0.005.

Figure 3.18: MSE of predictions on extrapolated **TLR**. The MSE between the predicted and simulated values. The MSE is calculated per pixel averaged over the timeseries.

4. Discussion

In this research a universal approach towards surrogate modelling for simulation models of geographical systems is explored. The ML algorithm used in this research is a RF-model. This RF model has the same configuration for each simulation it emulates during the study, to test the universality of this approach. The research starts with a proof-of-concept, where the RF model predicts one timestep of three different simulation models. The simulation models that are used are the game of life simulation, a forest fire model and a snow model. In the proof of concept stage is seen that the model can predict the forest fire model and game of life simulation with an accuracy of 99,8% and 98,7%, respectively. The predictions on the snow model were too high, but the RF model predicted the overall spatial pattern of snow.

Based on the promising results of the proof-of-concept stage, the research continued in the next stage, focusing on examining the impact of external inputs $\mathbf{I}(x, y, \text{t})$ and neighbourhoods $\mathbf{N}(\mathbf{S}(x, y, \text{t}))$ on the predictive performance of the forest fire and snow model. The results show that $\mathbf{I}(x, y, \text{t})$ and $\mathbf{N}(\mathbf{S}(x, y, \text{t}))$ are of great importance in emulating the simulated model. Not taking the $\mathbf{N}(\mathbf{S}(x, y, \text{t}))$ into account for the forest fire or disregarding precipitation and temperature in the snow model results in predictions that deviate only slightly around the starting point. These results answer the first and second sub-questions: *How does the RF model perform when it is trained exclusively on the state variable?* **and** *To what extent is the performance of the RF model dependent on neighbourhood interactions and external input such as precipitation and temperature?*

The research concluded by testing the RF model on its ability to perform on extrapolated and interpolated parameters of the simulation model. The performance of the RF model on interpolation and extrapolation differs substantially between simulation models. The RF model seems to perform well on interpolation for the parameter $\mathbf{R}$ and well on the extrapolation of both parameters of α and $\mathbf{R}$ for the forest fire model. These results answer the third sub-question: *How effectively does the RF model generalise to unseen data, in terms of interpolation and extrapolation?*

These findings combined offer a glimpse into the **Main research question: Can a universal RF model emulate different types of pixel based numerical models of geographical systems?** The universal RF model in its current configuration performed well in the proof of concept. It can emulate simpler simulation models, such as the simplified forest fire model used in this research. The RF model did not seem to capture the details of the snow model. However, these shortcomings can be improved by allowing for different configurations of the RF model.

It is important to acknowledge certain limitations within this study. The RF model configuration in this study is kept relatively simple. The maximum tree depth of 10 is unlikely to pick up on more complex patterns such as the patterns underlying the snow model. The study of a more complex, less transparent RF model as a universal surrogate can be an interesting addition to this research. An-

other limitation in this study is the diversity of simulation models used to test the universality of this approach. The scope of the research can be expanded by including a wider range of simulation models and geographical systems would improve the universality of the research. This could involve different spatial scales, different types of simulations, or additional variables of current simulations. Furthermore, relaxation of the transparency constraint would allow for the evaluation of different ML algorithms. Exploring the temporal dependencies in the simulation models by implementing a recurrent neural networks or to pick up on spatial dependencies without explicitly defining $\mathbf{N}(\mathbf{S}(x, y, \mathrm{t}))$, offers promising directions for future research.

Finally, in the forest fire models, the propagation and outline of the fire relies significantly on a nuanced ratio between the parameters α and $\mathbf{R}$. In future research, when assessing the performance of a universal surrogate model on parameter interpolation and extrapolation, it is important to maintain the ratio between these parameters instead of altering them individually. Modifying the parameters individually may lead to unrealistic outcomes.

5. Conclusion

An initial step has been taken towards implementing a ML-model as a universal surrogate in simulation models for geographical systems. Despite its limited configuration in this study, the RF model shows promising capabilities in emulating various simulation models, when external inputs and neighbourhood interactions are implemented in the training. While the current configuration of the RF model's is suitable for simpler simulation models, it demonstrates limitations in accurately emulating more intricate models such as the snow model.

Finally, it is important to consider the implications and potential applications of the findings in this research. The software developed in this research serves as a foundation for emulating any pixel-based model using an RF model. A first step is made in the direction of a general software framework for utilising ML-models as surrogate models in simulation models of geographical systems. By advancing this software to incorporate different ML models the software framework can further enhance its emulation capabilities and handle even more complex simulation models.

Bibliography

[1] J. Braun, P. Van Der Beek, P. Valla, *et al.*, "Quantifying rates of landscape evolution and tectonic processes by thermochronology and numerical modeling of crustal heat transport using pecube," *Tectonophysics*, vol. 524, pp. 1–28, 2012.

[2] J. L. Kavanagh, S. L. Engwell, and S. A. Martin, "A review of laboratory and numerical modelling in volcanology," *Solid Earth*, vol. 9, no. 2, pp. 531–571, 2018.

[3] P. Favreau, A. Mangeney, A. Lucas, G. Crosta, and F. Bouchut, "Numerical modeling of landquakes," *Geophysical Research Letters*, vol. 37, no. 15, 2010.

[4] R. Hassani, D. Jongmans, and J. Chéry, "Study of plate deformation and stress in subduction processes using two-dimensional numerical models," *Journal of Geophysical Research: Solid Earth*, vol. 102, no. B8, pp. 17 951–17 965, 1997.

[5] K. Kochanski, D. Rolnick, P. Donti, and L. Kaack, "Climate change+ ai: Tackling climate change with machine learning," in *AGU Fall Meeting Abstracts*, vol. 2019, 2019, GC33A–04.

[6] I. Karafyllidis and A. Thanailakis, "A model for predicting forest fire spreading using cellular automata," *Ecological Modelling*, vol. 99, no. 1, pp. 87–97, 1997, ISSN: 0304-3800. DOI: https://doi.org/10.1016/S0304-3800(96)01942-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304380096019424.

[7] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating physics-based modeling with machine learning: A survey," *arXiv preprint arXiv: 2003.04919*, vol. 1, no. 1, pp. 1–34, 2020.

[8] E. Goldstein, G. Coco, A. Murray, and M. Green, "Data-driven components in a model of inner-shelf sorted bedforms: A new hybrid model," *Earth Surface Dynamics*, vol. 2, no. 1, pp. 67–82, 2014.

[9] A. A. Kajbaf, M. Bensi, and K. L. Brubaker, "Temporal downscaling of precipitation from climate model projections using machine learning," *Stochastic Environmental Research and Risk Assessment*, vol. 36, no. 8, pp. 2173–2194, 2022.

[10] S. Rasp, M. S. Pritchard, and P. Gentine, "Deep learning to represent subgrid processes in climate models," *Proceedings of the National Academy of Sciences*, vol. 115, no. 39, pp. 9684–9689, 2018.

[11] M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, and N. Carvalhais, "Deep learning and process understanding for data-driven earth system science," *Nature*, vol. 566, no. 7743, pp. 195–204, 2019.

[12] K. Chattrairat, W. Wongseree, and A. Leelasantitham, "Comparisons of machine learning methods of statistical downscaling method: Case studies of daily climate anomalies in thailand," *Journal of Web Engineering*, pp. 1397–1424, 2021.

[13] G. Camps-Valls, D. Tuia, X. X. Zhu, and M. Reichstein, *Deep learning for the Earth Sciences: A comprehensive approach to remote sensing, climate science and geosciences*. John Wiley & Sons, 2021.

[14] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, "Machine learning–accelerated computational fluid dynamics," *Proceedings of the National Academy of Sciences*, vol. 118, no. 21, e2101784118, 2021.

[15] F. Hamilton, A. L. Lloyd, and K. B. Flores, "Hybrid modeling and prediction of dynamical systems," *PLoS computational biology*, vol. 13, no. 7, e1005655, 2017.

[16] N. D. Brenowitz, B. Henn, J. McGibbon, *et al.*, "Machine learning climate model dynamics: Offline versus online performance," *arXiv preprint arXiv:2011.03081*, 2020.

[17] K. Dagon, B. M. Sanderson, R. A. Fisher, and D. M. Lawrence, "A machine learning approach to emulation and biophysical parameter estimation with<? xmltex\break?> the community land model, version 5," *Advances in Statistical Climatology, Meteorology and Oceanography*, vol. 6, no. 2, pp. 223–244, 2020.

[18] P. A. O'Gorman and J. G. Dwyer, "Using machine learning to parameterize moist convection: Potential for modeling of climate, climate change, and extreme events," *Journal of Advances in Modeling Earth Systems*, vol. 10, no. 10, pp. 2548–2563, 2018.

[19] X. Wang, W. Xue, Y. Han, and G. Yang, "Efficient climate simulation via machine learning method," *arXiv preprint arXiv : 2209.08151*, 2022.

[20] S. Kawamleh, "Can machines learn how clouds work? the epistemic implications of machine learning methods in climate science," *Philosophy of Science*, vol. 88, no. 5, pp. 1008–1020, 2021.

[21] F. Regazzoni, L. Dede, and A. Quarteroni, "Machine learning for fast and reliable solution of time-dependent differential equations," *Journal of Computational physics*, vol. 397, p. 108 852, 2019.

[22] T. Qin, K. Wu, and D. Xiu, "Data driven governing equations approximation using deep neural networks," *Journal of Computational Physics*, vol. 395, pp. 620–635, 2019.

[23] J. H. Faghmous and V. Kumar, "A big data guide to understanding climate change: The case for theory-guided data science," *Big data*, vol. 2, no. 3, pp. 155–163, 2014.

[24] V. M. Krasnopolsky and M. S. Fox-Rabinovitz, "Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction," *Neural Networks*, vol. 19, no. 2, pp. 122–134, 2006.

[25] L. J. Slater, L. Arnal, M.-A. Boucher, *et al.*, "Hybrid forecasting: Blending climate predictions with ai models," *Hydrology and Earth System Sciences*, vol. 27, no. 9, pp. 1865–1889, 2023.

[26] P. Parisouj, H. Mohebzadeh, and T. Lee, "Employing machine learning algorithms for streamflow prediction: A case study of four river basins with different climatic zones in the united states," *Water Resources Management*, vol. 34, pp. 4113–4131, 2020.

[27] P. Stolfi and F. Castiglione, "Emulating complex simulations by machine learning methods," *BMC bioinformatics*, vol. 22, no. 14, pp. 1–14, 2021.

[28] S. Scher, "Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning," *Geophysical Research Letters*, vol. 45, no. 22, pp. 12–616, 2018.

[29] C. C. IPCC, *Mitigation of climate change. contribution of working 557 group iii to the sixth assessment report of the intergovernmental panel on climate 558 change*, 2022.

[30] H. ten Eikelder, *Utilize-machine learning to capture dynamics of large scale high resolution numerical models*, https://github.com/RicktenE/Utilize-Machine-Learning-to-Capture-Dynamics-of-Large-Scale-High-Resolution-Numerical-Models, 2023.

[31] *The nature of code*, https://natureofcode.com/book/chapter-7-cellular-automata/, (Accessed on 02/03/2023).

[32] A. Alexandridis, D. Vakalis, C. Siettos, and G. Bafas, "A cellular automata model for forest fire spread prediction: The case of the wildfire that swept through spetses island in 1990," *Applied Mathematics and Computation*, vol. 204, no. 1, pp. 191–201, 2008, ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2008.06.046. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0096300308004943.

[33] *1.10. decision trees — scikit-learn 1.2.2 documentation*, https://scikit-learn.org/stable/modules/tree.html, (Accessed on 06/23/2023).

[34] D. Karssenberg, O. Schmitz, P. Salamon, K. De Jong, and M. F. Bierkens, "A software framework for construction of process-based stochastic spatio-temporal models and data assimilation," *Environmental Modelling & Software*, vol. 25, no. 4, pp. 489–502, 2010.

[35] *Feature importances with a forest of trees — scikit-learn 1.2.2 documentation*, https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html, (Accessed on 06/20/2023).

[36] *4.2. permutation feature importance — scikit-learn 1.2.2 documentation*, https://scikit-learn.org/stable/modules/permutation_importance.html#:~:text=The%20permutation%20feature%20importance%20is,model%20depends%20on%20the%20feature., (Accessed on 06/20/2023).

[37] W. L. Fons, "Analysis of fire spread in light forest fuels," *Journal of Agricultural Research*, vol. 72, pp. 92–121, 1946.

[38] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN: 1441412697.

[39] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[40] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[41] PCRaster, *Software for environmental modelling*, Oct. 2010. [Online]. Available: https://pcraster.geo.uu.nl/.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[43] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[44] *Welcome to imageio's documentation! — imageio 2.31.1 documentation*, https://imageio.readthedocs.io/en/stable/, (Accessed on 06/23/2023).

6. Appendix A - Simulation models

6.1 Details on game of life

The game of life simulation aims to simulate if life as described by certain rules can live in a randomly initiated state of cells being alive or dead. the simulation works based on a set of algorithmic rules determining the state of each cell. Each cell can be classified as either alive or dead. The rules that dictate the evolution of the cells are as follows:

Birth: (Before birth a cell is considered dead). A dead cell comes to life in the next timestep, if exactly three cells in it's neighbourhood $\mathbf{N}$(S(x, y, t)) are alive. This rule represents reproduction, where a new cell is born due to favourable conditions.

Survival: A living cell continues to live on the next timestep if it has two or three neighbouring cells that are alive. This rule reflects the idea that a cell can sustain its existence when surrounded by enough support.

Death: A living cell dies in the next timestep if it has fewer than two neighbouring cells that are alive (underpopulation) or more than three neighbouring cells that are alive (overpopulation). These conditions represent that a cell cannot thrive due to insufficient support or excessive competition for resources. These rules are applied simultaneously to all cells at each timestep and described by the following update function,

$$\mathbf{S}_g(x,y,t+1) = \begin{cases} 1, & \text{if } \sum \mathbf{N}(S(x,y,t)) = 2 \text{ or } 3 \\ 0, & \text{if } \sum \mathbf{N}(S(x,y,t)) < 2 \text{ or } > 3. \end{cases} \tag{6.1}$$

Where $\mathbf{S}_g(x,y,t+1)$ represents the new state of the cell at time step t+1, $\sum \mathbf{N}(S(x,y,t))$ counts the alive neighbours. The update function follows the general shape of Eq. 2.1 without any $\mathbf{I}$(x, y, t) or $\mathbf{P}$(x, y). The neighbours at the edges are defined in such a way that it takes the value of the pixel on the other side of the grid.

6.2 Details on forest fire simulation

depending on the complexity of the model each cell can have more states. In the case of forest fires simulations the grid of the CA simulation represents a landscape in which a forest fire can take place or has taken place. The most important factors determining the spread of a fire are the vegetation type, wind, humidity, topography, fuel density and spotting [37]. In this study only the topographical data $\mathbf{I}(x, y, t)$, the output of the previous step $\mathbf{S}(x, y, t)$ and the neighbouring cells $\mathbf{N}(S(x, y, t))$ are used as features for the model. In this research a simplified model is used to simulate a forest fire [32].

6.2.1 Topography

The topography is an important factor, as an increase in temperature changes the density of the air surrounding the fire, the less dense air moves opposite gravity and rises, together with radiated heat from the fire, pre-heating the vegetation in at higher altitudes. The steeper the slope, the faster the fire spreads in that direction [6]. The probability of catching fire, determined by the topography surrounding the cell is given by:

$$p_t = \mathbf{R}e^{(\alpha\ \theta_e)} \tag{6.2}$$

Where $\mathbf{R}$ is the spread rate in a flat landscape, α determines the strength of the effect of the slope and θ_e refers to the slope between the cell and its neighbour. The calculations for determining the slope are shown in equation Eq. 6.3,

$$\theta_e = \begin{cases} \tan^{-1}(\frac{E_1 - E_2}{l}) & \text{if horizontal/vertical neighbour} \\ \tan^{-1}(\frac{E_1 - E_2}{l\sqrt{2}}) & \text{if diagonal neighbour} \end{cases} \tag{6.3}$$

Where $E_1 - E_2$ denotes the difference in elevation between the cell and its neighbour, l is the length of one cell. The probability of a cell changing state p_{burn} can then be described as,

$$p_{burn} = p_h * p_t. \tag{6.4}$$

Where p_h is a constant probability of a cell catching fire (when a neighbouring cell is on fire) and p_t is the probability given by the topography (flat or curved).

6.3 Details on snow model

6.3.1 Temperature lapse rate (TLR)

For the duration of this simulation, each day the temperature was measured at a meteorological station at a height of 2058.1m. One day is presented by one timestep in the simulation, $t \rightarrow t + 1$. The temperature for the other cells in the map, $\mathbf{T}(x, y,$ t), is calculated based on this temperature and the relative elevation to the meteorological station and the Temperature Lapse Rate (**TLR**), as described in the equation below,

$$\mathbf{T}(x, y, t) = \mathbf{T}(x_{ms}, y_{ms}, t) + (\mathbf{E}(x, y) - \mathbf{E}(x_{ms}, y_{ms})) * \mathbf{TLR}. \tag{6.5}$$

Where $\mathbf{T}(x, y,$ t) represents the temperature on each cell at each timestep, $\mathbf{E}(x, y)$ represents the elevation of each cell and x_{ms}, y_{ms} represent the coordinates of the meteorological station. Note that this calculation happens at time t, this calculation occurs for each cell at each timestep once. Also note that **TLR** is constant during the complete simulation, it is initially set to 0.005 [$^{\circ}\mathrm{C\,km^{-1}}$].

6.3.2 Defining snowfall and melt

The model uses $\mathbf{T}(x, y,$ t) to determine if the precipitation is snow or rain.

$$\mathbf{P}(x, y, t) = \begin{cases} \mathbf{P}(x, y)_{SNOW}, & \text{if } \mathbf{T}(x, y, t) < 0.0 \\ \mathbf{P}(x, y)_{RAIN}, & \text{if } \mathbf{T}(x, y, t) \geq 0.0 \end{cases} \tag{6.6}$$

The model also takes into account how much snow potentially melted and flowed out of the cell. The amount of melted snow is calculated as a function of $\mathbf{T}(x, y,$ t), where the following linear relationship is assumed between $\mathbf{T}(x, y, t)$ and the melt rate,

$$\mathbf{M}(x, y, t) = \mathbf{T}(x, y, t) * K \tag{6.7}$$

Where $\mathbf{M}(x, y,$ t) is the amount of melted snow at time t, and K is the rate set to 0.01 [m $^{\circ}\mathrm{C}^{-1}$]. The direction in which the melted snow streams follows a local drain direction map. This local drain direction map is derived from the topographical data and is not on by itself an input for the model. This simulation model has two dynamic drivers $\mathbf{I}(x,y,$t), temperature and precipitation, and one static input $\mathbf{P}(x, y)$, the topographical data.

7. Appendix B - Software stack

The software used during this research is listed below.

Primary software:
Python version(3.10)[38]

For data manipulation:
numpy version(1.23.5)[39]
pandas version(1.5.3)[40]

For the forest fire simulation and snow accumulation models:
PCRaster version(4.4.0) [41]

For the random forest model, performance measure and some visualisations:
scikit-learn version(1.2.1)[42]

For visualisations and animations:
matplotlib version(3.6.3)[43]
imageio.v2 version(2.25.0)[44]